METHOD AND SYSTEM FOR AUTOMATED TECHNICAL SUPPORT FOR COMPUTERS

TECHNICAL FIELD

This patent application relates in general to the field of computing devices, and more particularly to a method and system for automating support for computers.

BACKGROUND

Personal computer systems have become increasingly common in businesses and households. Although the term "personal computer" implies a generic device, "personal computers" generally have a wide diversity of hardware and software components. For instance, different personal computers may have processors and buses of different speeds, hard drive and RAM memories of different sizes, and peripheral devices interfaced with different types of interface cards, such as audio devices. Further, a large array of manufacturers produce computer components so that in a given personal computer, even components having substantially similar operating characteristics may have important differences based on each component's manufacturer specification.

With respect to software, generally all personal computers have a common need for an operating system that coordinates the operation of hardware components. However, each individual personal computer may have one of many possible operating systems. For instance, Microsoft products have evolved from its original Disk Operating System ("DOS") to Windows systems, including Windows 3.1, Windows 95, Windows 98, Windows CE and Windows NT. In addition to these Microsoft operating systems, other types of operating systems are available, such as different versions of Unix, including Linux.

In addition to this wide diversity of operating systems, personal computers may operate a large number of different types of software applications. A given software application may interact in different manners with different operating

systems. Thus, even with substantially similar hardware components, personal computers having different software may operate in substantially different manners.

Computer users can experience difficulties in system operation for many reasons. Lack of knowledge, hardware faults, software incompatibilities, and many other causes can lead to problems for the computer user. Given the wide range of hardware and software available (which implies an even greater range of hardware/software combinations that a user can experience), it is difficult to determine if the computer has a problem.

This situation is further complicated by the fact that personal computers do not have good mechanisms to automatically determine if the hardware/software system is having a problem. While certain operating systems contain code that help sense some types of problems with specific pieces of hardware, such mechanisms may be insufficiently uniform for determining if the operating system has a problem. Indeed, a common symptom of an operating system problem is a failure to boot, in which case the OS cannot be counted on to help. Another common symptom of an operating system problem is a hang, in which case the operating system becomes unresponsive to the keyboard and mouse for a wide variety of possible causes. It should be noted that this type of problem can be caused by pieces of software which have been installed on top of the operating system, such as an application or driver, or some incompatibility between pieces of software that have been loaded. A system that was operational may stop functioning at some later point due to software incompatibilities.

Another issue is the lack of a uniform mechanism for the user to invoke assistance. If the user has a question or the system has a problem, or at least the user perceives a problem, there is currently no uniform mechanism to get the system to attempt to provide assistance to the user. Although there are various types of help available to the user, they rely on one or more working input devices, such as a mouse and/or a keyboard, and a sufficient level of user knowledge to be able to navigate to one of a variety of information sources on the system and on a global information source such as the internet.

SUMMARY

Therefore, a need has arisen for a method and system for identifying and resolving personal computer system problems that is accessible through a uniform fail-safe mechanism regardless of the functional state of the operating system and other software, and can be implemented on a wide variety of operating systems

A further need exists for a method and system that detects when an operating system has failed to boot or has hung and can take appropriate corrective actions.

A further need exists for such a system that includes a monitoring system that communicates with the operating system and vice versa, and that is capable of doing so with a wide variety of different operating systems.

A further need exists for a standard mechanism that can be invoked to attempt to resolve operating system failure to boot and operating system hang conditions.

A further need exists for such a standard mechanism that will attempt to resolve operating system hang conditions regardless of whether the assistance has been requested by the user during boot or otherwise, and regardless of whether the user has made multiple requests for assistance.

In accordance with the present disclosure, a method and system is provided that substantially eliminates or reduces disadvantages and problems associated with previously developed methods and systems for identifying computer system problems. A monitoring system detects problems with a computer system and aides in identifying and resolving the problems. The current level of functionality of the computer system is determined, and technical support is provided for the computer system in accordance with the functionality of the computer system.

According to one aspect of the invention, a state machine monitors operating system functionality to detect computer system failures. A watchdog timer is initiated substantially simultaneously with initiation of computer system boot and cleared at a predetermined point of the computer system boot sequence. A computer system boot failure is determined to exist if the watchdog timer remains uncleared after a predetermined time period. For instance, the watchdog timer is cleared with an operating system service routine before expiration of the predetermined time period, thus indicating that the operating system has booted through the service routine point of the boot sequence within the predetermined time period. Failure to clear the

watchdog timer with the service routine indicates failure of the boot process through the boot sequence point at which the service routine is called.

In one embodiment, a user initiates operating system monitoring by pressing a service button to indicate a problem with the computer system. The pressing of the service button initiates support functions, such as the initiation of a service application, at an appropriate time. The support functions allow testing of the computer system by the monitoring system. The service button initiates a watchdog timer associated with monitoring the boot through calling of the operating system. Alternatively or in addition to the initiating of the watchdog timer associated with monitoring the boot, the service button initiates another watchdog timer that acts as a as a hang detection timer. If the service button is pressed during computer system boot, the hang detection timer is initiated at a predetermined point of the computer system boot sequence, such as after a user provides log in information, and is cleared upon initiation of the service application. An operating system hang-up error is identified if the hang detection timer remains uncleared after a predetermined hang detection time.

According to one embodiment, detection of a computer failure results in a reboot of the computer system to a service mode. The service mode boots up a service mode operating system to enable analysis of the computer system even if the computer system's primary operating system has failed. Initiation of a service mode boot also starts a watchdog timer. The watchdog timer is cleared at a predetermined point of the service mode operating system boot sequence. A computer system failure is determined to exist if the watchdog timer remains uncleared after a predetermined time period. If the service mode boot was initiated by a previous user press of the service button and ensuing fault detection, then a service mode hang detection timer monitors the service mode operating system boot sequence to detect any hang-up of the service mode operating system.

In yet another embodiment, a method for automated support is provided in a computer system having a service button and a controller chip set. The method includes the steps of pressing the service button, setting a first bit in a general purpose input register in the controller chip set to generate a first interrupt signal in response to the pressing step, receiving the first interrupt and determining whether the

computer system is booting, and if the system is booting then initiating a service application routine in a first manner, or if the system is not booting then initiating a service application routine in a second manner.

A computer system is also provided having a processor with at least one timer, a controller chip set, a system BIOS, and an operating system for communicating with components of the computer system through the BIOS. A service button is coupled with a general purpose input register in the chip set for setting the register for generating a first interrupt. The system further includes an interrupt handler that is coupled to the input register for receiving the first interrupt and processing it in a manner dependent on whether the computer system is in a booting state or a non-booting state.

A computer system having a system BIOS and an operating system is also provided, wherein the computer system includes a service button coupled to a general purpose input register in a controller chip set for setting a bit in the register to generate a first interrupt signal. An interrupt handler in the system BIOS receives the first interrupt signal and initiates a second interrupt signal to the operating system to initiate a service application if the computer system is not in a booting state. If the computer is in a booting state the bit remains set, and code contained within the operating system checks the status of the bit later during the booting sequence and initiates the service application if the bit has been set.

The present invention provides many important technical advantages. One important technical advantage is integrated support for detecting problems associated with computer systems. Monitoring a computer system boot sequence for hardware or operating system failure enables automation of problem detection and support for resolving the problems. Further, detection of an operating system failure allows analysis and correction of the computer system problem through use of the service mode operating system.

Another important technical advantage is the automatic confirmation that a problem exists with a computer system. Indication that the monitoring system has detected a problem provides, as a minimum, confirmation to technical support staff with reduced dependence on the verbal description of the computer system user. Problem confirmation limits the number of basic items that technical support staff

need to check over the course of a telephone call. Further, if the monitoring system does not detect a problem, then technical support staff can limit the number of problems needing investigation. For instance, failure to detect a problem with the monitoring system indicates that the hardware and operating system have booted in a normal manner, and the system is capable of initiating the service application.

Another important technical advantage is the identification of the problem associated with the computer system. For instance, monitoring of the computer system boot allows identification of problems as associated with hardware or with the operating system, or alternatively may indicate proper hardware and operating system functionality that indicates user or application related difficulties. If operating system software is the problem, use of a service mode operating system supports full analysis to further identify and analyze the problem. For instance, if the main operating system is inoperable, then the service mode operating system supports computer system operation and allows operation of the computer system for automatic analysis and correction of the problem with the main operating system.

Another important advantage is a robust user interface that is simple and uncomplicated to use. For instance, a user with a question or problem simply pushes a single service button. Pressing the service button generates an interrupt directly into the chip set to alert the monitoring system that service is requested by the user. The direct interface of the service button to the chip set enhances reliability and simplicity, as the user's input to the service button does not have to rely on the operation of computer components, such as a keyboard or mouse. Additionally, a user may press the service button at any time to seek assistance. The means by which pressing of the service button initiates a service application ensures that the service application will be run at the appropriate time, regardless of when the service button is pressed, and regardless of whether it is pressed multiple times. Once the service button is pressed, the computer system may perform an indepth analysis of potential problems, even when the operating system has failed, by using the service mode operating system to run computer components. Further, the system and method of the present invention can be easily implemented with a variety of different operating systems.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of the present invention and advantages thereof may be acquired by referring to the following description taken in conjunction with the accompanying drawings, in which like reference numbers indicate like features, and wherein:

FIGURE 1 depicts a block diagram of a computer system monitored by an operating system monitor state machine;

FIGURE 2 depicts a flow diagram of operating system monitoring during and after a normal mode boot;

FIGURE 3 depicts a flow diagram of operating system monitoring during and after a service mode boot;

FIGURE 4 depicts a flow diagram of initiation of a service application following pressing of a service button; and

FIGURE 5 depicts a block diagram of software and hardware elements used to initiate a service application.

DETAILED DESCRIPTION

Preferred embodiments of the present invention are illustrated in the Figures, like numeral being used to refer to like and corresponding parts of the various drawings.

A healthy operating system monitors hardware and software operations on a computer system. At times, the operating system detects difficulties or problems with the computer system and provides notice of the difficulties or problems to the computer system user. Help systems generally associated with the operating system can help to resolve difficulties or problems either automatically or through user interaction, such as by asking questions. However, when the operating system itself has a problem or there are software incompatibilities, it is difficult for the operating system to address those problems. Frequently, the operating system either shuts down or hangs up without providing further notice of the problem to the computer system user.

To improve computer system problem detection, identification and resolution, a monitoring system associated with the computer system's BIOS monitors operating system functionality. The monitoring system detects operating system boot failures and various types of operating system hang-ups. Once a problem is detected, remedial action is automatically taken to recover a failed computer system using a uniform mechanism that takes advantage of operational aspects of the computer system. In addition, the monitoring system can be invoked by a single push of a service button. Pushing of a service button provides an interrupt to the computer system chip set for automatically invoking the highest available level of user assistance as determined by computer system health and state. As will be described more fully below, the service button may be pressed by a user while the computer system is in POST, booting, in service mode or in normal mode. When the service button is pressed, the BIOS sets a bit in a general purpose input register in the controller chip set and generates an interrupt. State-sensitive interrupt handler code in BIOS takes appropriate action, and may communicate to the operating system, depending upon the state of computer system as represented by certain CMOS bits. Further, the interrupt handler code ensures that only appropriate action is taken regardless of the number of times the service button is successively pressed. A

system and method for monitoring computer system failure will now be described in more detail, followed by a detailed description of the manner in which the service button can invoke such a monitoring system.

Referring now to FIGURE 1, a block diagram depicts a computer system 10 having an operating system 12 interfaced with hardware components 14 through a basic input output system ("BIOS") 16. Hardware components 14 include conventional personal computer system hardware components such as a processor, modem, audio card, video card, and storage devices, including a hard drive, floppy drives, ROM and RAM. On initial power-up or upon initiation of a reboot, BIOS 16 directs a boot sequence, including a power on self test ("POST") and calling of the operating system. Within hardware 14 resides one or more timers 18 and 19, such as conventional watchdog timers.

BIOS 16 boots computer system 10 on power up in a conventional manner. A monitor state machine 20 monitors the boot process by comparing state transitions through the boot sequence against expected results. Monitor state machine 20, for instance, communicates with timer 18 to compare the expected time for a predetermined transition from a first point of the boot sequence to a second point against elapsed time for the sequence. If timer 18 expires uncleared, then a problem has been detected based upon the timer's expiration. If BIOS 16 successfully boots computer system 10 to bring operating system 12 on line, then a service routine of operating system 12 clears timer 18 to prevent an indication of a problem.

If monitor state machine 20 detects a problem with computer system 10, BIOS 16 may direct a number of different responses. For instance, BIOS 16 may call up a service mode operating system with service protocol 22. The service mode operating system may, for instance, be a simplified version of operating system 12, such as the Windows' Safe mode for Windows 98. The service mode operating system may include a modem driver so that the computer system can contact an analysis server through the Internet to upload user symptoms, system configuration and state information, as well as to run automated analysis software and diagnosis. BIOS 16 also may illuminate service lights 24 to indicate detection of a problem, with a different configuration of lights indicating the identity of one or more specific problems. A computer user may then provide the illuminated light information to

technical support to help analyze and solve the problem. Alternatively, technical support may obtain system information from the analysis server.

Computer system 10 includes a service button 26 available for a computer user to press. Service button 26 provides a robust user interface that enables a user to initiate the problem detection and identification process. As is described further below, service button 26 generates an interrupt into the computer system chip set to, for instance, initiate a service application. Monitor state machine 20 detects pressing of the service button and launches the service application or monitors the system behavior to detect computer system problems.

In addition to monitoring the boot sequence through the calling of operating system 12, monitor state machine 20 may monitor the functioning of operating system 12 with a hang detection timer 19. If the service button has been pressed during boot, hang detection timer 19 is initiated during the boot sequence, for instance, by a user login, and cleared by an application run after completion of the calling up and booting of operating system 12 or service mode operating system 22. If the application does not clear hang detection timer 19 within a predetermined period of time, monitoring state machine 20 determines that an operating system hang-up has occurred. BIOS 16 then recognizes an operating system problem and attempts a service mode boot or indicates a probable hardware failure through service lights 24.

Referring now to FIGURE 2, a flow diagram depicts steps for support automation for operating system monitoring of a normal boot mode. At step 50, a normal computer boot is initiated. For instance, the user of computer system 10 may apply power or may have instructed the operating system to reboot the system. At step 52 a hang detection watchdog timer is initiated. The booting of the operating system and the activity of the timer proceed in parallel within the system. The watchdog timer counts down. If it reaches zero before steps 58 and 60 are completed (i.e., when a service routine runs late in the boot process and clears the timer), then step 54 is achieved, and the system is rebooted into service mode in step 56.

Typically, the boot sequence tests hardware and initiates operating system boot within a predictable time period. Upon completion of the hardware testing, such as the POST test, and initiation of the operating system boot, an instruction is sent from an operating system service routine to clear the watchdog timer at step 58. If the

watchdog timer is cleared at step 60, a normal boot is indicated. If the watchdog timer is not cleared and counts down to zero, the process proceeds to step 56 for rebooting into service mode with the service mode operating system. In one embodiment, additional normal boots may be automatically repeated before proceeding to the service mode boot sequence. In summary, if the watchdog timer remains uncleared after a predetermined period of time then it is known that the computer system did not boot through the point of the boot sequence at which the service routine clears the watchdog timer. Thus, the problem with the computer system may be identified to a certain degree based upon the boot sequence that was or was not completed.

Step 58 indicates that an operating system service routine is run at a predefined point in the latter part of the computer boot process. At step 60, the watchdog timer is cleared with the operating system service routine before expiration of the watchdog timer predetermined time period. If step 60 is reached, then the computer hardware and software tested and run up to initiation of the predefined point of the operating system boot sequence should generally be operational. Once this determination is made, at step 62 a user is provided an opportunity to login.

At step 64, a determination is made of whether the service button 26 was pressed during the OS boot process (see further discussion below), as opposed to a normal boot in which the service button was not pressed. If the determination is no, the process proceeds to step 70 for initiation of normal computer system operation.

If at step 64 it is determined that pressing of the service button occurred during boot, then at step 66, a service application is initiated and at step 72 a hang detection timer is initiated for monitoring of an operating system hang-up. Hang detection monitoring uses hang detection timer 19 or another timer to test whether the operating system completes its service application launch within a predetermined period of time. The hang detection timer is initiated at step 72 and then cleared by an application running on the computer system after the application completes a predetermined portion of its loading and startup sequence at step 68. Thus, at step 68 a determination is made of whether the application loading and startup sequence is normal. If yes, the application running on the computer system clears the hang detection timer and the process proceeds to step 70 for initiation of support

applications. If the hang detection timer remains uncleared for a predetermined period of time, then at step 74 a determination is made that the service application has failed to clear timer 19. This may indicate that the operating system has hung; at a minimum, it is incapable of starting the service application normally. At step 74, upon detection of expiration of the hang detection timer, the system is rebooted into normal mode (step 50) or service mode (step 76), dependent upon a definable number of unsuccessful attempts to reboot into normal mode (step 75).

If a user presses the service button while the computer is in normal operation, such as at step 70 of FIGURE 2, or at anytime other than boot of the computer, then the system proceeds to step 78 to test the operating system for a hang-up. A service application is initiated at 66 and the hang detection timer is initiated at step 72. If the service application clears the timer at step 68, then the computer system proceeds to normal operation at step 70. If the timer expires at step 74, then an operating system hang-up is detected and the system attempts to reboot to normal mode (at step 75) until a specifiable number of normal reboot failures have occurred, at which point the system reboots to service mode at step 76. This allows a determination of operating system functionality without a need for a complete reboot. Further, if the timer expires at step 74, trouble shooting is possible with the service mode even though the normal operating system is not functional.

Referring now to FIGURE 3, service mode is initiated at step 80 with a service mode boot sequence. At step 82, a service mode watchdog timer is initiated. As above, this watchdog timer counts down toward zero in parallel with the loading of the (in this case, service mode) operating system. If the watchdog timer reaches zero (step 84) before it is cleared late in the service mode boot process (steps 88 and 90), then the service mode boot has failed and this is indicated by setting the LEDs to indicate hardware failure (step 86). A hardware problem is probable since neither the main operating system nor the service mode operating system were able to bring the computer to an operational state.

At step 88, a service mode operating system routine is run at a predefined point in the later part of the boot process. If the service routine runs at step 88, then at step 90 the routine clears the timer to indicate that the service mode operating system

is functional. At step 92, computer system boot is completed with the service mode operating system.

At step 94, a determination is made of whether the service button was pressed during the normal mode or service mode boot. The state of button pressed during boot is stored across reboot attempts that fail. If yes, then the service application is launched and a system hang detection test proceeds with initiation of the operating system hang detection timer at step 104. Again, the counting down of the hang detection timer occurs in parallel with the loading and startup of the service mode recovery application at step 96. If this succeeds, it will run code to clear the hang detection timer at step 98. At this point, the service mode operating system is known to be at least functional enough to start the service mode application. At step 100, service support applications are initiated such as those useful for analyzing software problems. The computer system is operating in service mode and is available for troubleshooting.

At step 106, if the watchdog timer counts down to zero (i.e., expires) prior to being cleared by the service application, then at step 108 the service mode operating system has been shown to be incapable of loading and starting the service application within the preset time period. At this point, a hang of the service mode operating system has been detected, and the process ends with indication of a probable hardware problem on lights associated with the computer system at step 108.

At step 102, if the service button was pressed while the computer is in service mode operation, then the system proceeds to steps 104 and 96 in parallel to test the service mode operating system for a hang-up. A service application is initiated at step 96 and the hang detection timer is initiated at step 104 If the service application clears the timer at step 98, then the computer system proceeds to service mode operation at step 100 to allow initiation of service mode recovery applications that allow analysis of the computer system failure and corrective action. If the timer expires at step 106, then an operating system hang-up is detected and the system indicates a probable hardware failure at step 108. This allows a determination of service mode operating system functionality without a need for a complete reboot.

The following case examples will further clarify the operation of the monitoring system. If the monitoring system finds no hardware or operating system

faults, then the computer user may seek a resolution of the problem or question through local help on the computer system or by connecting with help over the internet using the computer system. Local and remote internet-based help will resolve the majority of computer problems or questions.

Another helpful case example is a non-fatal hardware failure, such as a CD-ROM or audio speaker card failure. The monitoring system should indicate that no failure has occurred with the operating system, and the user may contact technical support to have new hardware sent. Some types of non-fatal hardware failures will limit the options available for obtaining help. For instance, the computer system will operate without a modem or network interface card ("NIC"). However, failure of this hardware will limit the computer system's ability to interact over the Internet to obtain help. In part, modem failure may be addressed by entering the service mode. For instance, if modem failure is associated with modem configuration or ISP dial instructions, then a service mode modem configuration may support an Internet-based problem solution.

In another case example, if the normal mode operating system is not operational, will not boot or is otherwise unstable, a modem connection may be established with the service mode operating system. The Internet connection through the service mode allows direct system analysis of the operating system to automatically support operating system problem resolution and operating system restoration. For instance, a new operating system or relevant parts of the operating system may be loaded over the Internet to replace the faulty operating system. If the automatic problem resolution fails to resolve the problem, then the user may call technical support and identify the problem based on the displayed light configuration.

As one additional case example, the computer system may have a fatal flaw that prevents operation in both the normal and the service modes. For instance, the computer system may be set up incorrectly or may have a fatal hardware fault, such as a motherboard, hard drive or power supply fault. In such instances, an explanatory chart provided with the computer system will provide the problem associated with display light configurations, and simple instructions for the user to follow. The user may then use this information to contact technical support and obtain replacement hardware.

As indicated above, computer system 10 includes a service button 26 available for a computer user to press. Service button 26 generates an interrupt into the computer system chip set to, for instance, initiate a service application. Monitor state machine 20 detects pressing of the service button and launches the service application at an appropriate time, or monitors the system behavior to detect computer system problems. When the service button is pressed, hang detection timer 19 is initiated, and is later cleared by an application run after completion of the calling up and booting of operating system 12 or service mode operating system 22. If the application does not clear hang detection timer 19 within a predetermined period of time, monitoring state machine 20 determines that an operating system hang-up has occurred. BIOS 16 then recognizes an operating system problem and initiates a predetermined reboot protocol that may include rebooting in service mode as is described in detail above.

The service button provides a standard mechanism through which a user can invoke assistance. Referring now to FIGURES 4 and 5, a user who seeks to invoke assistance will press service button 26 at step 400. Note that although not specifically shown, the flow chart depicted in FIGURE 4 involves two execution spaces, one within the BIOS and the other within the operating system execution space. In general, communication to the operating system is handled by generating an interrupt, such as a system control interrupt (SCI), while communication from the operating system back to the BIOS is accomplished by running code that sets values in the BIOS, such as clearing a hang detection timer. The means by which the monitoring system in the BIOS communicates with the operating system and the operating system responds (if not hung), as will be more fully described below, provides unique advantages. Although the system is of necessity operating-system dependant since some portions reside within the operating system, it is also capable of leveraging the underlying personal computer architecture to allow the same mechanism in the BIOS to support multiple operating-system specific implementations. Further, the system enables user assistance to be invoked regardless of the functional state of the operating system.

As shown in FIGURE 5, the service button 26 is wired directly to a specific input register 500 in the general purpose input/output register (GPIO) of the controller

chip set 520, and pressing of the service button causes a bit in that input register to be set at step 402. The setting of this bit generates a system management interrupt (SMI) at step 404 to initiate state-sensitive interrupt handler code, an SMI handler 502, in the BIOS. The SMI handler 502 receives the SMI, and disables further SMI generation at step 406 until the present SMI has been serviced to ensure that if a user presses the service button multiple times, only one interrupt will be generated until that interrupt has been fully serviced.

At step 408, the SMI handler determines whether the computer system is booting by examining the appropriate bit in the CMOS register. If the system is currently booting, the general purpose input bit remains set while the system continues its boot sequence. A hang detection timer is also set at step 410, but the SMI handler takes no further action. When the system completes its boot sequence, or at a predetermined point in the booting sequence where it is known that hardware and software tested and run up to that point in the boot sequence are generally operational, such as when the user is prompted for a login ID, the operating system is directed to check the status of the service button bit at step 411. If the service button bit has been set, indicating that the button was pressed during boot, the operating system will launch a service application at step 422, otherwise it will resume normal operations (step 412). In one embodiment, a background task such as a service application launcher associated with the operating system, which is run as part of the normal boot process, checks the service button bit. If the service button bit is set, the service application launches the service application.

If, at step 408, the SMI handler determines that the system is not booting, the SMI handler initiates a hang detection timer at step 416. This hang detection timer could be the same timer as would be set in step 410 above or a different timer. The value to which the timer is set, however, will be different depending on whether the service button was pressed during boot or otherwise. If pressed during boot it will be set to a higher value, representing the longer amount of time required to allow the system to complete the boot cycle and launch the service application. If not pressed during boot, the timer will be set to a lower value, representing the shorter amount of time required to allow the system to process the interrupt (described below) and start the service application.

If the system is not booting, the SMI handler code in the BIOS subsequently communicates with the operating system by causing an interrupt at step 418 to notify the operating system that the service button has been pressed. In one embodiment, this interrupt is a system control interrupt (SCI) that is serviced in operating system execution space. To initiate the SCI, the SMI handler sets an output bit 504 in an output register in the GPIO. As shown in FIGURE 5, this bit is used as an input to a system control interrupt input 506, which in turn initiates the SCI 508. At step 420, the SCI is processed by an interrupt service routine (ISR) 510 in the operating system execution space. The ISR provides a message to the operating system to initiate a service application. In one embodiment this is achieved by sending a message to the service application launcher 512 associated with the operating system, which starts the service application 514 at step 422.

Regardless of whether the service button was pressed during boot or otherwise, if the service application starts correctly, as determined at step 426, the service button bit and hang detection timer are cleared at step 428. In one embodiment, the service application notifies the service application launcher and instructs it to clear the service button bit and the hang detection timer. If the service application has not started correctly (the timer has reached zero before being cleared), it may indicate an operating system hang-up or, at minimum, that it is incapable of properly starting the service application. Thus, at step 430, the system begins to follow a predetermined reboot protocol possibly including rebooting in service mode, as is described in detail above. Finally, once the SMI has been fully serviced the SMI handler reenables SMI generation at step 432 so that subsequent pressing of the service button will cause another interrupt and initiate servicing as described above.

Thus, the system and method of the present invention provide a unique way in which to invoke user assistance in a uniform fail-safe way. The manner in which the code in the BIOS execution space communicates with the operating system and vice versa enables invocation of a service request that is operating system independent, and provides a monitoring system that is outside of the operating system so as to be able to monitor the operating system itself. Further, the system and method described above enables a user to invoke assistance regardless of the state of the operating system (i.e. during booting or otherwise, or when the operating system is hung).

The problem identification and resolution systems described herein may be provided as a build-to-order component of the computer system. For instance, less sophisticated users may order their computer system with single-button push problem resolution, whereas more experienced users may order computer systems with standard configurations. Alternatively, computer system buyer's may order only portions of the system, such as a timer associated with just main operating system monitoring that does not include the ability to automatically call up a service mode operating system.

Although the present invention has been described in detail, it should be understood that various changes, substitutions and alterations can be made hereto without departing from the spirit and scope of the invention as defined by the appended claims.



CLAIMS:

1. A method for monitoring a computer system boot sequence, the method comprising:

initiating a timer;

clearing the timer if a predetermined point of the computer system boot sequence occurs; and

determining that a computer system failure exists if the timer remains uncleared after a predetermined time period.

2. The method of Claim 1, wherein said clearing step further comprises clearing the timer with an application run by said computer system.

3. The method of Claim 2, further comprising initiating a reboot of the computer system with a second operating system when said determining step determines that a computer system failure exists.

4. The method of Claim 3 further comprising, following the step of initiating a reboot:

initiating a timer;

clearing the timer with an application run by the computer system if a predetermined point of the computer system reboot sequence occurs; and

determining a service mode operating system failure if the timer remains uncleared after a predetermined time period.

5. A computer system comprising:

a processor having at least one timer;

a BIOS for booting the computer system;

an operating system for supporting computer system operations; and

a monitoring state machine associated with the BIOS and in communication with the processor, the monitoring state machine detecting operating system failure by comparing elapsed time for an operating system function against a predetermined time period, the elapsed time measured with the timer.



6. The system of Claim 5 further comprising a service operating system, the monitoring state machine operational to call the service operating system upon detecting an operating system failure.

7. The system of Claim 5 further comprising a service button, wherein pressing of said service button interrupts the processor and initiates the timer.

8. A method for providing automated technical support for a computer system, comprising the steps of:

pressing a service button at any time during operation of said computer system, said pressing of said service button causing an application for providing said automated technical support to be invoked regardless of the operating state of said computer system.

9. The method of Claim 8, wherein said service button is coupled to a chip set, and wherein said service button invokes said application by causing said chip set to generate at least a first interrupt.

10. The method of Claim 9, wherein said causing step further comprises the steps of:

setting a first bit in a general purpose input register in the controller chip set to generate a first interrupt in response to said pressing step;

receiving said first interrupt and determining whether said computer system is executing a booting sequence; and

if said computer system is not booting, initiating a service application routine in a first manner; or

if said computer system is booting, initiating said service application routine in a second manner at a predetermined time during said booting sequence.

11. The method according to claim 10, further comprising the step of initiating a timer in response to said pressing step, and clearing said timer if said

service application reaches a predetermined point before said timer reaches a predetermined value.

12. The method according to claim 11, further comprising the step of initiating a reboot of the computer system if said service application has not reached said predetermined point prior to said timer reaching said predetermined value.

13. The method according to claim 11, wherein said step of initiating said service application in said second manner further comprises the steps of:

checking the status of a predetermined bit of said chip set at a predetermined point during the booting sequence; and

if said predetermined bit has been set, initiating said service application.

14. The method according to claim 13, wherein said step of initiating said service application in said first manner further comprises the step of generating a second interrupt, said second interrupt initiating an interrupt service routine, said interrupt service routine initiating said service application.

15. The method according to claim 14, wherein said second interrupt is a system control interrupt, and said step of initiating said system control interrupt further comprises the step of setting a second bit in a general purpose output register of said controller chip set, said setting of said second bit causing the generation of said system control interrupt.

16. A computer system comprising:

a chip set;

a memory;

an application for providing automated technical support for said computer; and

a service button for invoking said application at any time during operation of said computer system regardless of the operating state of said computer system.

17. The computer system according to claim 16, wherein said service button is coupled with said chip set for generating at least a first interrupt when said service button is pressed to invoke said service application.

18. The computer system according to claim 17, further comprising:

a processor having at least one timer;

a system BIOS;

an operating system for supporting computer system operations and for communicating with components of said computer system through said BIOS;

wherein said service button is coupled with a general purpose input register in said chip set for setting said register to generating said first interrupt;

an interrupt handler comprising code in the system BIOS, said interrupt handler being coupled with said input register for receiving said first interrupt and processing said interrupt in a manner dependent on whether the computer is in a booting state or a non-booting state.

19. The computer system according to claim 18, wherein said interrupt handler determines if said computer system is in said booting or said non-booting state, said interrupt handler being coupled with a general purpose output register in said chip set for setting a bit in said output register if said computer system is in said non-booting state, and wherein said general purpose output register is coupled with a third register in said chip set, said third register generating an interrupt signal for initiating a second interrupt when said bit in said output register is set.

20. The computer system according to claim 19, wherein said second interrupt calls an interrupt service routine in said operating system that initiates said service application.

21. A computer system according to claim 20, wherein said interrupt handler is coupled to said timer for initiating said timer substantially with receiving said first interrupt.